Journal of Machine Learning Research 26 (2025) 1-40 Submitted 1/24; Published 1/25

Memory Gym: Towards Endless Tasks to Benchmark Memory Capabilities of Agents

Marco Pleines MARCO.PLEINES@TU-DORTMUND.DE
Department of Computer Science
TU Dortmund University
Dortmund, 44227, Germany

Matthias Pallasch MATTHIAS.PALLASCH@TU-DORTMUND.DE
Department of Computer Science
TU Dortmund University
Dortmund, 44227, Germany

Frank Zimmer FRANK.ZIMMER@HOCHSCHULE-RHEIN-WAAL.DE
Department of Communication and Environment
Rhine-Waal University of Applied Sciences
Kamp-Lintfort, 47475, Germany

Mike Preuss M.PREUSS@LIACS.LEIDENUNIV.NL
Leiden Institute of Advanced Computer Science
Universiteit Leiden
EZ Leiden, 2311, The Netherlands

Editor: Martha White

Abstract

Memory Gym presents a suite of 2D partially observable environments, namely Mortar Mayhem, Mystery Path, and Searing Spotlights, designed to benchmark memory capabilities in decision-making agents. These environments, originally with finite tasks, are expanded into innovative, endless formats, mirroring the escalating challenges of cumulative memory games such as "I packed my bag". This progression in task design shifts the focus from merely assessing sample efficiency to also probing the levels of memory effectiveness in dynamic, prolonged scenarios. To address the gap in available memory-based Deep Reinforcement Learning baselines, we introduce an implementation within the open-source CleanRL library that integrates Transformer-XL (TrXL) with Proximal Policy Optimization. This approach utilizes TrXL as a form of episodic memory, employing a sliding window technique. Our comparative study between the Gated Recurrent Unit (GRU) and TrXL reveals varied performances across our finite and endless tasks. TrXL, on the finite environments, demonstrates superior effectiveness over GRU, but only when utilizing an auxiliary loss to reconstruct observations. Notably, GRU makes a remarkable resurgence in all endless tasks, consistently outperforming TrXL by significant margins.
Website and Source Code: `https://marcometer.github.io/jmlr_2024.github.io/`

Keywords: deep reinforcement learning, actor-critic, memory, transformer, recurrence

1. Introduction

Imagine embarking on a long-awaited family vacation, with the open road stretching ahead. As the car hums along, a lively atmosphere envelops the passengers, young and old alike, as they engage in a playful game of "I packed my bag". Each family member takes turns adding an item to an imaginary bag, attempting to remember and recite the growing list correctly. However, as the game unfolds and the list lengthens, the passengers encounter mounting challenges illustrating the finite nature of their individual memories. Despite best efforts, varying strategies, and different capabilities, the weight of the growing list eventually leads to resignation.

Memory is not just a game – it is a critical tool for intelligent decision-making under imperfect information and uncertainty. Without the ability to recall past experiences, reasoning, creativity, planning, and learning may become elusive. In the realm of autonomously learning decision-making agents as Deep Reinforcement Learning (DRL), the agent's memory involves maintaining a representation of previous observations, a knowledge bank that grounds its next decision. Memory mechanisms, be it through recurrent neural networks (Rumelhart et al., 1986) or transformers (Vaswani et al., 2017), have enabled these agents to master tasks both virtual and real. For instance, DRL methods have conquered complex video games as Capture the flag (Jaderberg et al., 2018), StarCraft II (Vinyals et al., 2019), and DotA 2 (Berner et al., 2019). Their success extends beyond virtual environments to real-world challenges as dexterous in-hand manipulation (Andrychowicz et al., 2020) and controlling tokamak plasmas (Degrave et al., 2022). These non-Markovian examples underscore that effective memory mechanisms are crucial; without them, the tasks are fundamentally unsolvable as they require the robust maintenance and manipulation of information over time.

1.1 Endless Tasks Benchmark Memory Effectiveness and not just Efficiency

To discover the most effective memory approach for complex tasks, as mentioned earlier, insightful benchmarks are needed. However, conventional DRL memory benchmarks, which typically revolve around finite tasks that terminate upon reaching a terminal state regardless of success or failure, fall short in providing the necessary depth. Finite tasks impose a predetermined limit on the demands placed on an agent's memory. When various memory strategies successfully complete a fixed task, their performance is typically assessed based on efficiency rather than effectiveness. Here, "efficiency" refers to the number of agent-environment interactions required to achieve effective behaviors, commonly known as sample efficiency. Although these methods may reach the same end goal, their true effectiveness and memory capabilities often remain uncertain. For example, a memory mechanism that retains more information or sustains it over extended periods could prove more effective, potentially leading to higher scores, increased returns, better survival times, or improved win rates. This distinction could identify it as a more robust and effective approach compared to others assessed.

Cumulative memory games, such as "I Packed My Bag" and "Simon"(Morrison and Baer, 1977), overcome the limitations of finite tasks by introducing an automatic curriculum through their inherently progressive and unbounded nature. These games represent a special instance of endlessness: while theoretically offering a truly endless challenge, in practice,

episodes inevitably terminate due to the finite nature of memory. The game's expansion relies on the agents' continuous success, thereby embodying the concept of "endlessness" in tasks. This approach provides deeper insights into the effectiveness of decision-makers when facing escalating demands for memory retention and recall. Consequently, these challenges rigorously test an agent's memory, evaluating both the quantity of information it can retain and the duration for which it can robustly maintain it.

1.2 Contributions: Novel Memory Benchmark and Transformer-XL Baseline

To exploit the endless behavior of cumulative memory games to thoroughly benchmark memory effectiveness, we enhance our prior work Memory Gym (Pleines et al., 2023), an open-source benchmark, designed to challenge memory-based DRL agents to memorize events across long sequences, generalize, be robust to noise, and be sample efficient. Memory Gym encompasses three distinct finite environments: Mortar Mayhem, Mystery Path, and Searing Spotlights. Each environment presents visual observations, multi-discrete action spaces, and crucially, they cannot be mastered without memory, demanding frequent memory interactions. Building on this foundation, we advance Memory Gym by evolving each environment into an endless format, inspired by the "I packed my bag" game, to further test and develop memory capabilities in DRL agents. To our understanding, no existing DRL memory benchmark possesses such endless tasks.

We further contribute an easy-to-follow baseline implementation of Transformer-XL (TrXL) (Dai et al., 2019) to the open-source CleanRL library (Huang et al., 2022b), based on the widely used Deep Reinforcement Learning (DRL) algorithm Proximal Policy Optimization (PPO) (Schulman et al., 2017). A recurring theme in the realm of memory-based DRL is the lack of accessible and reproducible baselines. Many significant works, particularly those utilizing transformers (Fortunato et al., 2019; Parisotto et al., 2020; Hill et al., 2021; Lampinen et al., 2021; Parisotto and Salakhutdinov, 2021), have not made their implementations publicly available. As implementation details of DRL algorithms are vital (Engstrom et al., 2020; Andrychowicz et al., 2021; Huang et al., 2022a), this lack of transparency poses significant challenges for the research community, hindering progress and reproducibility. Therefore, our accessible implementation can serve as a beacon for the community fostering further research.

In our experimental analysis, we benchmark a TrXL and a recurrent agent, based on the Gated Recurrent Unit (GRU) (Cho et al., 2014), across the original (finite) and the new endless environments of Memory Gym. Our findings indicate that TrXL, only with the help of observation reconstruction as auxiliary loss, is most effective on the finite environments. To our greatest surprise, in the endless environments, GRU consistently surpasses TrXL by large margins.

1.3 Overview

This paper begins with related work, followed by illustrating the dynamics in Memory Gym's environments, particularly focusing on their endless formats. We detail the actor-critic model architecture, including the loss functions and key elements of the TrXL baseline. The subsequent section presents our experimental analysis, showcasing results from both finite and endless tasks. We further explore TrXL's underperformance in endless tasks,

providing insights and suggestions for future research. Before concluding, we falsify our initial hypothesis about recurrence's vulnerability to spotlight perturbations in Searing Spotlights, emphasizing the significant impact of not normalizing estimated advantages during agent optimization.

2. Related Work

Previous studies have explored the use of memory-based agents in a variety of tasks. Some of them are originally fully observable but are turned into partially observable Markov Decision Processes by adding noise or masking out information from the agent's observation space (Hausknecht and Stone, 2015; Heess et al., 2015; Meng et al., 2021; Shang et al., 2021). In the next subsection, we give a coarse overview of recently used benchmarks chronologically. Subsequently, we discuss how these benchmarks primarily assess efficiency, highlighting a limitation in their scope, which we address with our contribution of Memory Gym's endless environments.

2.1 An Overview of Memory Benchmarks in Deep Reinforcement Learning

Deepmind Lab 30 (Beattie et al., 2016) presents a collection of 30 procedurally generated first-person 3D environments. Starting from a general motor-control navigation task and visual observations, each environment poses a different goal such as collecting fruit, playing laser tag or traversing a maze. Pašukonis et al. (2023) have identified that agents can exploit the environments' skyboxes, which reduces the demands on memory.

Minigrid (Chevalier-Boisvert et al., 2018) includes a 2D grid memory task inspired by the T-Maze (Wierstra et al., 2007). In this environment, the agent is required to memorize an initial goal cue, traverse a long alley, and correctly choose the exit once the end is reached.

Miniworld (Chevalier-Boisvert, 2018) encompasses a set of first-person 3D environments sharing similarities to the tasks introduced by Minigrid.

VizDoom (Wydmuch et al., 2019) features first-person shooter 3D environments that revolve around motor-control navigation tasks, with the added challenge of computer-controlled enemies that may confront the agent.

The **Memory Task Suite** (Fortunato et al., 2019) includes diverse memory-based environments across four categories: PsychLab (Leibo et al., 2018) for image-based tasks (e.g. detect change), Spot the Difference for cue memorization, goal navigation tasks inspired by the Morris water maze (D'Hooge and De Deyn, 2001), and transitive object ordering tasks.

Procgen (Cobbe et al., 2020) consists of 16 2D environments that offer procedurally generated and fully observable levels. These environments encompass a combination of puzzle-solving, platforming, and action-based games. 6 of these environments have been modified to become partially observable by reducing and centering the agent's visual observation.

Numpad (Parisotto et al., 2020) requires the agent to accurately press a series of keys

in a specific sequence of fixed length. The sequence is not provided to the agent, resulting in the agent using a trial-and-error approach while memorizing the underlying sequence.

Memory Maze (1) (Parisotto et al., 2020) shares similarities with the Morris Water maze (D'Hooge and De Deyn, 2001), as the agent must find an apple, then undergoes random repositioning, requiring it to relocate the apple's location.

Dancing the Ballet (Ballet) (Lampinen et al., 2021) presents a sequence of up to 8 dances visually, with the agent remaining stationary. After all the dances are displayed, the agent is required to identify a specific one to successfully complete the task.

PopGym (Morad et al., 2023), published at ICLR 2023 alongside Memory Gym (Pleines et al., 2023), is a benchmark with 15 environments classified as diagnostic, control, noisy, game, or navigation. These are designed for rapid convergence, offering vector instead of visual observations.

Memory Maze (2) (Pašukonis et al., 2023) showcases procedurally generated 3D mazes, where agents, from a first-person perspective, are tasked with locating specific objects as requested by the environment. Additionally, the authors have collected a dataset tailored for offline reinforcement learning.

2.2 Conventional Benchmarks are Limited to Measuring Efficiency

All of the aforementioned environments conclude with terminal states, regardless of whether the agent succeeds or fails. Navigational tasks such as those found in Deepmind Lab 30, Minigrid, Miniworld, VizDoom, Procgen, Numpad, Memory Maze (1), and Ballet end by the agent discovering an exit. This closure also applies to the navigation challenges within the Memory Task Suite and PopGym. Additional tasks in PopGym conclude after the agent plays through an entire deck of cards or reaches a distinct goal state in games such as Concentration, Battleship, and Minesweeper. Memory Maze (2) and tasks in Psychlab terminate after a predefined number of steps.

Due to these clearly finite episodes, the demands on the agent's memory have an upper bound. Once the agent solves a given task, their performance is primarily measured by their efficiency. Efficiency here refers to the number of steps an agent requires to complete an episode. In scenarios such as escaping a maze, a more efficient agent will use fewer steps, whereas even a random agent might eventually find the exit if given sufficient time. In both the literature and our results on the finite environments of Memory Gym (Section 5.3), agents are typically compared based on the number of samples (i.e., agent-environment interactions) they use during training to achieve their performance. But what if an agent's memory is stronger at recalling larger amounts of relevant information over extended horizons? Efficiency metrics may fail to fully reveal the true memory capabilities of agents, including their capacity to retain larger amounts of information or sustain that information over extended periods.

While the related works still offer potential for scaling in terms of memory demands, we take inspiration from cumulative memory games to expand Memory Gym to provide endless

tasks that automatically and boundlessly test memory capabilities. Identifying particularly strong memory mechanisms may enable the tackling of complex tasks, as noted in the introduction.

3. Endless Memory Tasks Inspired by "I Packed My Bag"

Memory Gym's environments offer a unique feature that sets them apart from the related benchmarks: an endless design tailored to rigorously test memory capabilities. This section first explores how exploiting the concept of cumulative memory games can serve as a valuable benchmark to assess memory effectiveness, not merely efficiency. We then describe how we expand Memory Gym's finite environments into endless ones.

It is important to clarify that 'endless' should not be confused with 'open-ended'. Memory Gym's endless environments are not open-ended in nature; they possess clear objectives and defined structures. In contrast, open-ended environments offer agents freedom to develop unique skills and strategies without stringent constraints as demonstrated by the Paired Open-Ended Trailblazer algorithm (Wang et al., 2020) or platforms like Minecraft (Fan et al., 2022).

3.1 Evaluating Effectiveness through Cumulative Memory Games

As argued in Section 2.2, measuring sample efficiency may obscure the true effectiveness of an agent's memory capabilities, especially under the assumption that the evaluated baselines are sufficient to solve the given finite tasks. If endless tasks are not considered, what alternative methods can be used to assess an agent's memory effectiveness? There are two primary approaches: scaling the difficulty of the task or scaling the agent.

One approach involves progressively ablating the agent itself while maintaining a fixed task difficulty. For instance, the dimensions of transformer architectures or recurrent cells are scalable in terms of their parameter count, which is indicative of the memory's capacity. By iteratively reducing these dimensions, a threshold is eventually reached beyond which the agent can no longer perform effectively. Thus, with a fixed task, we can only determine the minimal configuration that still allows the agent to be effective.

In contrast, incrementally increasing the task's difficulty can determine the point at which an agent fails, thereby exposing the agent's limits of effectiveness. The field of Curriculum Learning (CL), a specialization of transfer learning, offers methods to automate this scaling process (Narvekar et al., 2020). CL enhances training efficiency by sequentially transferring knowledge from simpler tasks to more challenging ones, ultimately addressing tasks that might be too complex to tackle from scratch. While setting up curricula could introduce implementation overhead, such as hand-crafting tasks or scheduling transitions between tasks, cumulative memory games naturally function out-of-the-box as an automatic curriculum. These endless games increase in difficulty as the agent progresses within an episode, using a trivial heuristic to adjust the challenge in an online fashion, thereby eliminating the need for a separate task scheduler.

Applying the concept of cumulative memory games to a DRL memory benchmark provides significant insights. Agents that excel at these tasks demonstrate robustness in retaining and recalling vast amounts of information over extended periods. Moreover, because these games offer unbounded difficulty, they remain appropriately challenging—neither too

difficult nor too easy—and hence continue to provide the right challenge for evaluating future memory mechanisms. Mastery of these memory-intensive games highlights an agent's suitability for real-world applications, where maintaining context and adapting to incremental information are essential.

3.2 From Finite to Endless: Expanding Memory Gym's Environments

Prior to the underlying work, we introduced Memory Gym, a set of finite tasks benchmarking DRL agents to strongly depend on frequent memory interactions (Pleines et al., 2023). Memory Gym consists of the environments Mortar Mayhem, Mystery Path, and Searing Spotlights. All of them expose visual observations, capturing an area of 84×84 RGB pixels, to the agent and allow for multi-discrete actions. Leveraging procedural content generation, each episode is restarted with a new level sampled from a distinct set of seeds. Additionally, the environments' simulation speed measures multiple thousands of steps per second, which is faster than most related benchmarks (see Appendix F). Originally, these tasks were designed with finite goals, which limited the scope of memory challenges presented to the agents.

In transforming these environments from finite to endless, we aim to enhance the challenge posed to an agent's memory by increasing the amount of information to memorize over expanding time:

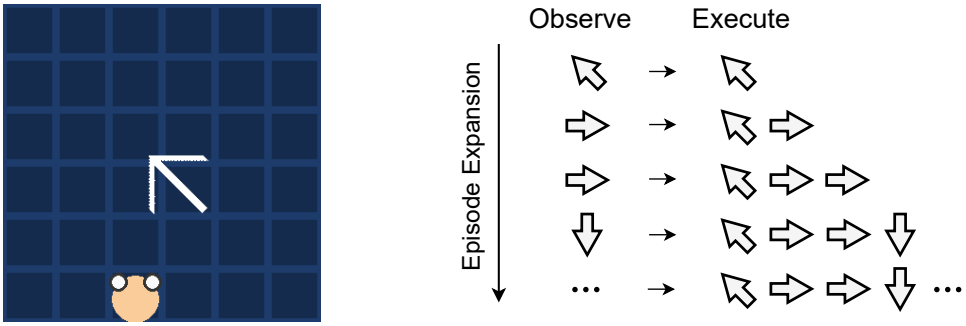
- **Endless Mortar Mayhem** tasks the agent to memorize and execute an ever-expanding list of commands in the correct order.
- **Endless Mystery Path** exhibits a never-ending, invisible path for the agent to traverse. Deviating from the path forces a restart from the beginning, requiring the agent to recall both the route and previous missteps accurately.
- **Endless Searing Spotlights** demands the agent to track its obscured location while avoiding threatening spotlights and collecting coins.

For detailed configuration and scaling of these environments, refer to Annex A, which presents the reset parameters. The next subsections explore the transitions from finite tasks to endless ones.

3.2.1 Endless Mortar Mayhem

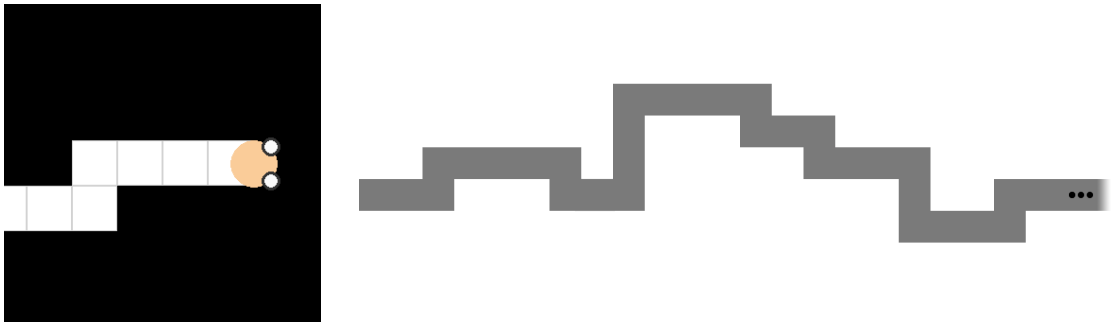
The finite variant of Mortar Mayhem takes place inside a grid-like arena and comprises two tasks. Initially, the agent is immobile and must memorize a sequence of ten commands, followed by executing each command in the observed order. A command instructs the agent to move to an adjacent floor tile or remain at the current one. Failure to execute a command results in episode termination, whereas each successful execution yields a reward of $+0.1$.

Endless Mortar Mayhem (Figure 1) extends the to-be-executed command sequence continuously, enabling potentially infinite sequences. To accommodate this, the command visualization and execution are alternated. As shown in Figure 1(b), an episode begins by displaying one command, followed by its execution. Subsequently, the next visualization reveals only the second command, requiring the agent to execute both the first and second



(a) Agent Observation (b) Command Observation and Execution

Figure 1: Endless Mortar Mayhem (a) is observed from a top-down view and its arena occupies the entire screen. If the agent, depicted as a circle with white paws, walks off, it re-enters from the opposite side. Episodes endlessly expand by alternately visualizing and executing commands (b), behaving as an automatic curriculum.



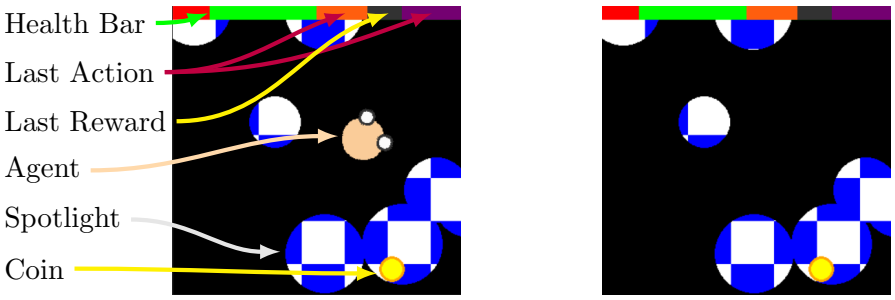
(a) Agent Observation (b) Endless Path

Figure 2: In Endless Mystery Path (a), the agent observes only a segment of the environment at a time. By rendering the path tiles past it, the agent acquires a sense of horizontal movement, even though its horizontal position remains visually constant. Figure (b) showcases an example of a procedurally generated endless path.

commands. This automatic curriculum allows the to-be-executed command sequence to continually expand, while each new command is presented only once.

3.2.2 Endless Mystery Path

The core concept of the original Mystery Path environment requires the agent to navigate an invisible path until reaching its end. If the agent strays off the path, it is relocated to the path's origin. To succeed, the agent must consistently memorize and recall its steps along the path, as well as the locations where it fell off.



(a) Annotated Ground Truth (b) Agent Observation

Figure 3: In Endless Searing Spotlights' annotated ground truth, the top rows of pixels display the agent's remaining health points, its last action, and indicate whether a positive reward was received during the last step. The last action is encoded using two chunks and three colors, while the last positive reward is represented by two colors. In the agent's observation (b), the spotlights play a role in revealing or hiding other entities.

Endless Mystery Path (Figure 2) introduces a never-ending path, which is always generated from left to right. Consequently, we eliminate the agent's ability to move left. As the endless path cannot be fully captured in an 84×84 RGB pixel observation, we visually fix the agent's horizontal position. If nothing but the agent is shown in the observation, the agent lacks information about its horizontal motion. To address this, we render the path tiles behind the agent, providing visual cues of the local horizontal position.

Regarding terminal conditions, we aim to shorten episodes if the agent performs poorly. Firstly, the agent is given a time budget of 20 steps to reach the next tile. This budget is reset when the agent moves to the next tile or falls off. Another condition terminates the episode if the agent falls off before reaching its best progress. Lastly, the episode is terminated if the agent falls off at the same location for a second time. With these conditions in place, an episode can potentially last indefinitely if the agent consistently succeeds.

In terms of rewards, the agent earns $+0.1$ for each previously unvisited tile of the path it reaches. While this may initially appear as a dense reward function, any misstep that causes the agent to fall off means it cannot reclaim those rewards. The agent must retrace its steps and regain its prior progress to continue accumulating rewards. The longer it takes to catch up, the sparser the rewards become.

3.2.3 Endless Searing Spotlights

Endless Searing Spotlights (Figure 3) features a pitch-black environment illuminated only by moving and threatening spotlights. The agent starts with ten health points, losing one each time it is hit by a spotlight. At the beginning of each episode, the environment is fully lit, but it quickly dims to darkness within a few steps. The episode ends if the agent's health points are depleted. To survive, the agent must cleverly use the darkness as cover, relying on memory to remember past actions and previous positions to infer its current

location. The agent's observation space includes its last action, current health, and any positive reward received in the previous step.

To mitigate monotonous behaviors, a coin collection task is incorporated. In the finite version, the agent aims to collect a randomly placed coin, worth +0.25, and then exit the level for an additional +1 reward. Effective memory use enables the agent to track its location, coin positions, and the exit. In the endless concept, the exit is removed, and the coin collection mechanism is revamped to spawn a new coin each time one is collected. This coin remains visible for six frames, while the agent has 160 steps to collect it. Distinct from the finite version, spotlights now appear at a steady rate, rather than intensifying over time.

4. Transformer-XL Baseline

In our experiments, we employ the widely-recognized Deep Reinforcement Learning (DRL) algorithm Proximal Policy Optimization (PPO) (Schulman et al., 2017). To add memory abilities to PPO, either recurrent neural networks (RNN) or attention mechanisms (i.e. transformer) are potential solutions. Prior works have extensively demonstrated the effectiveness of leveraging RNNs, such as Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho et al., 2014), within the realm of DRL (Mnih et al., 2016; Espeholt et al., 2018; Huang et al., 2022a). This also applies to previous studies employing transformers, as Gated Transformer-XL (GTrXL) (Parisotto et al., 2020) and Hierarchical Chunk Attention Memory (HCAM) (Lampinen et al., 2021). However, integrating these memory-enhancing techniques into DRL algorithms is not straightforward, particularly due to unavailable transformer-based DRL implementations. To bridge this gap, we contribute an easy-to-follow baseline implementation of Transformer-XL (TrXL) (Dai et al., 2019) in PPO. Before delving into the transformer-based PPO baseline, we describe the actor-critic model architecture and the losses used during training.

4.1 Actor-Critic Model Architecture

Figure 4 presents a broad overview of the actor-critic model architecture. The encoding process begins with visual observations, and optionally, vector observations (i.e. game state information). To encode visual observations (i.e. pixel observations) of shape $84 \times 84 \times 3$, we employ the Atari CNN (Mnih et al., 2015). The vector observation is encoded by a fully connected layer (FC). The outputs of these encoders are concatenated and subsequently embedded into the input space of the transformer-based memory encoder. Once the data is propagated through the memory encoder, it is further processed by each individual head that may contain further fully connected hidden layers. The policy head (actor) samples multi-discrete actions as required by Memory Gym, while the value head (critic) approximates the state-value.

Optionally, there is a head dedicated to reconstructing the original visual observation to strengthen the learning signal: the latent output of the memory encoder is fed to a transposed Atari CNN. The utility of such an auxiliary head is demonstrated by some of our results (Section 5) and prior works by Lampinen et al. (2021) and Hill et al. (2021). During development, we also tried to attach the reconstruction head directly to the encoder, but this led to catastrophic results. Another optional head establishes a diagnosis tool

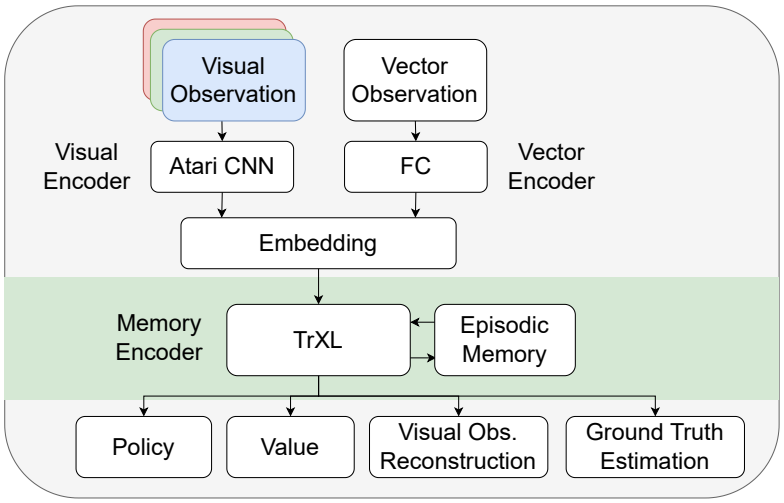


Figure 4: Overview of the actor-critic model architecture. The model features observation encoders and a memory encoder (green). The encoders' parameters are shared among several heads dedicated to the policy, state-value function, observation reconstruction, and ground truth estimation.

by estimating ground truth information, which is provided as labels by the environment. Despite the injection of illegal information into the agent's training, we utilize this head in Section 5.6.2 to demonstrate that TrXL's performance improves in Endless Mortar Mayhem due to this richer learning signal, indicating that its overall low performance is not attributed solely to a lack of capacity. Initially, the parameters of such a head can be optimized post-agent training to evaluate whether distinct information can be queried from the agent's memory (Baker et al., 2020).

4.2 Loss Function Composition

Our baseline utilizes PPO's clipped surrogate objective (Schulman et al., 2017). Due to leveraging memory, the to be selected action a_t of the policy π_θ depends on the current observation o_t and the memory encoder's output that we refer to as hidden state h_t.

$$L_t^C(\theta) = -\mathbb{E}_t\left[\min\left(q_t(\theta)A_\pi^{\text{GAE}}(o_t,h_t,a_t),\ \text{clip}\left(q_t(\theta),1-\epsilon,1+\epsilon\right)A_\pi^{\text{GAE}}(o_t,h_t,a_t)\right)\right] \quad (1)$$

$$\text{with ratio } q_t(\theta) = \frac{\pi(a_t|o_t,h_t,\theta)}{\pi(a_t|o_t,h_t,\theta_{\text{old}})}$$

A_π^{GAE} denotes advantage estimates based on generalized advantage estimation (GAE) (Schulman et al., 2016), θ the trainable parameters of a neural net, and ϵ the clip range. t depicts the current time step. The value function is optimized using the clipped squared-error loss $L_t^V(\theta)$, which is apparent in the implementation of Schulman et al. (2017):

$$L_t^{VClip}(\theta) = \left(\text{clip}\left(V(o_t,h_t,\theta), V(o_t,h_t,\theta_{\text{old}})-\epsilon, V(o_t,h_t,\theta_{\text{old}})+\epsilon\right) - \hat{V}_t\right)^2 \quad (2)$$

$$L_t^V(\theta) = \max\left[\left(V(o_t, h_t, \theta) - \hat{V}_t\right)^2,\ L_t^{VClip}(\theta)\right] \tag{3}$$

The auxiliary observation reconstruction is optimized using the binary cross entropy:

$$L_t^R(o_t, \hat{o}_t) = -[\hat{o}_t \log(o_t) + (1 - \hat{o}_t)\log(1 - o_t)], \tag{4}$$

and the optional ground truth estimation leverages the squared-error loss, which optimizes the concerned head's parameters ϕ:

$$L_t^Y(\hat{y}_t, y_t, \phi) = (\hat{y}_t - y_t)^2 \tag{5}$$

The final combined loss is depicted by $L_t^{C+V+H+R+Y}(\theta)$:

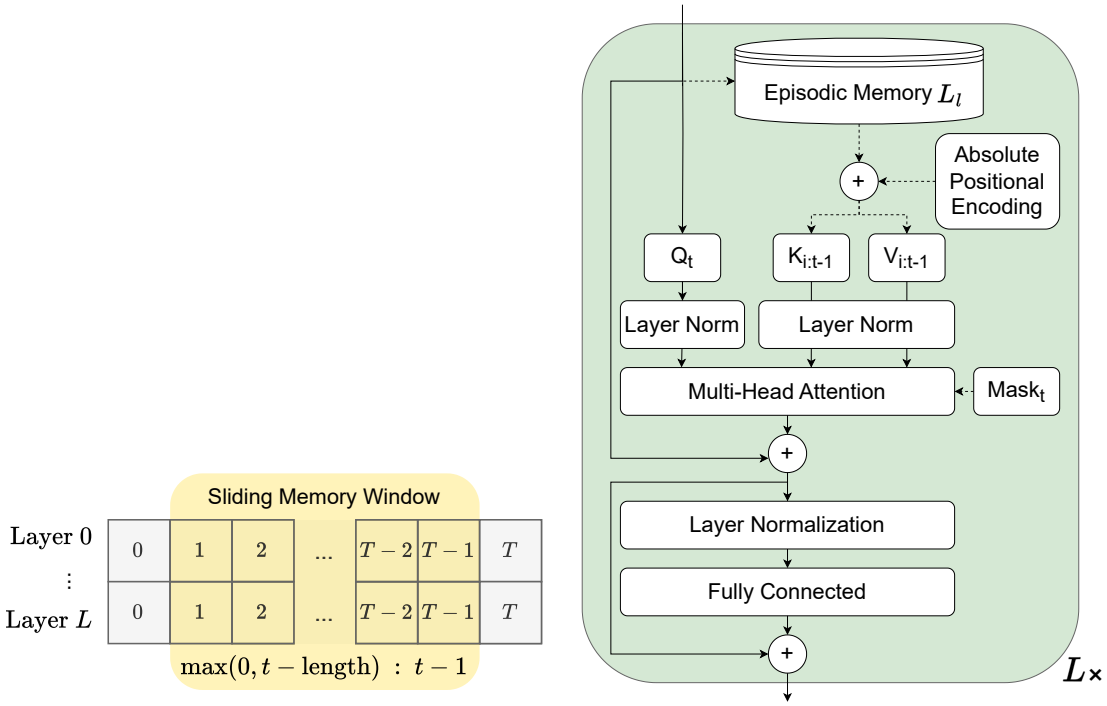
$$L_t^{C+V+H+R+Y}(\theta) = \mathbb{E}_t[L_t^C(\theta) + c_1 L_t^V(\theta) - c_2\mathcal{H}[\pi_\theta](o_t, h_t) + c_3 L_t^R(o_t, \hat{o}_t) + c_4 L_t^Y(\theta)] \tag{6}$$

where $\mathcal{H}[\pi_\theta](o_t, h_t)$ denotes an entropy bonus encouraging exploration (Schulman et al., 2017). c_1 to c_4 are coefficients to scale each loss, excluding the clipped surrogate objective.

4.3 Transformer-XL as Memory Mechanism

When processing sequential data, transformers employ attention mechanisms to capture global dependencies in parallel across the entire sequence (Vaswani et al., 2017), while RNNs iteratively propagate information through recurrent connections. Our transformer baseline draws conceptual inspiration from TrXL (Dai et al., 2019), GTrXL (Parisotto et al., 2020), and HCAM (Lampinen et al., 2021). The original transformer architecture proposed by Vaswani et al. (2017) is designed as a sequence-to-sequence model with an encoder-decoder structure. However, for the purpose of DRL, we adapt the architecture to a sequence-to-one model, focusing solely on the encoder as done in GTrXL and HCAM. This modification is driven by the underlying Markov Decision Process, which maps the environment's current state to a single action rather than a sequence of actions.

To begin with, it needs to be clarified what kind of sequential data is fed to a transformer, which is typically composed of multiple stacked layers. Naively, the input data can be conceptualized as a sequence of observations. In the context of visual observations, this is commonly referred to as frame stacking, where consecutive frames are combined. However, instead of expensively stacking raw frames, we leverage the past (i.e. cached) outputs of the model's embedding module (Figure 4), which serves as the input sequence to the first transformer layer. The subsequent layers make use of the past outputs from their preceding layers. These sequences, those elements we refer to as cached hidden states, are stored in the agent's episodic memory (Figure 5(a)) and are collected during inference based on the episode's current timestep t. To accommodate computational challenges of potentially endless episodes, we employ a sliding memory window approach, where the input sequence of these hidden states is derived from a fixed-length window. By leveraging cached hidden states in the input sequence, the agent's memory facilitates the ability to attend to events that extend beyond the boundaries of the fixed-length memory window, aligning with the



(a) Episodic Memory

(b) Transformer-XL Layer

Figure 5: Figure (a) illustrates the episodic memory that stores past inputs to all Transformer-XL layers for every step taken by the agent. Input sequences to the Transformer-XL layers are retrieved using a sliding memory window. T denotes the episode length. Figure (b) depicts the architecture of the Transformer-XL block that stacks L layers. The dashed lines state that there is no gradient flow into the episodic memory, the absolute positional encoding, and the mask.

segment-level recurrence of TrXL (Dai et al., 2019). Let N represent the number of layers and L denote the window length. The maximum context length L_{max} is given by

$$L_{\text{max}} = N \times (L - 1) + 1. \tag{7}$$

Note that gradients are not back propagated through the episodic memory because of caching. Once propagated through all transformer layers, a final hidden state is produced to inform the agent's policy.

The architecture of our TrXL encoder block is depicted in Figure 5(b). At each timestep t, the block receives an input that is solely based on the current timestep. This input is added to the episodic memory and serves as the query Q_t during the execution of multi-head attention (MHA) (Vaswani et al., 2017). The keys K and values V used in MHA are derived from the same data based on self-attention. Specifically, we retrieve K and V

by slicing the episodic memory using the bounding indices $i = \max(0, t - \text{window length})$ and $t - 1$. To ensure the positional information of K and V remains coherent, we add a positional encoding following the approach from Vaswani et al. (2017). We use the absolute variant as a default as we found a learned one to be effective, but less sample efficient. The underlying positional encoding matrix is scaled to the maximum episode length and not to the length of the window. Therefore, every sequence element of an episode is associate with its distinct position. In the case of endless episodes, the length of the positional encoding is fixed to 2048.

To restrict attention to time steps up to the current timestep t, we employ a strictly lower triangular matrix as a mask in MHA. This mask is only necessary when t is smaller than the length of the sliding memory window. The remaining configuration of the block adheres to the findings of Parisotto et al. (2020), which suggest improved performance with pre layer normalization and the identity map reordering. We encountered vanishing gradient issues when applying layer normalization after MHA and the fully connected layer, which is referred to as post layer normalization. Although our implementation supports the gating mechanism of GTrXL, it resulted in either lower or equivalent performance while being computationally more expensive (Appendix D).

5. Experimental Analysis

We run empirical experiments on Memory Gym using the just introduced TrXL baseline and one based on GRU, both powered by Proximal Policy Optimization (Schulman et al., 2017). The next subsection explains why GRU and TrXL are the only baselines considered. Next, we detail the evaluation protocol. Section 5.3 presents results on the finite tasks, showing that TrXL exhibits stronger policies than GRU, but only when leveraging observation reconstruction as an auxiliary loss. Contradicting our initial expectations and the results on the finite environments, we unveil surprising findings that highlight GRU's superiority over TrXL in all endless environments. Afterward, we compare the results from the finite and endless environments, demonstrating that the finite settings offer only a limited view of the agents' true memory capabilities. We then broadly explore several hypotheses on why TrXL might be less effective. Finally, we falsify that recurrent agents are vulnerable to spotlight perturbations when using enhanced hyperparameters (e.g., not normalizing advantage estimates).

5.1 Considered Baselines

Other related and significant baselines, such as the Memory Recall Agent (MRA) (Fortunato et al., 2019), Hierarchical Chunk Attention Mechanism (HCAM) (Lampinen et al., 2021), and Gated Transformer-XL (GTrXL) (Parisotto et al., 2020), are not considered due to their closed-source nature, which hampers reproduction. Unlike our previous study (Pleines et al., 2023), we exclude HELM (History comprEssion via Language Models) (Paischer et al., 2022a) and its successor, HELMv2 (Paischer et al., 2022b). This decision stems from HELM's suboptimal wall-time efficiency, which is six times more expensive than that of a GRU agent (Pleines et al., 2023), making hyperparameter optimization impractical for our purposes. Given our strict compute budget of 25,000 GPU hours, we dedicated our resources to developing the endless environments and the TrXL baseline instead.

For a comprehensive overview of the hyperparameters used in our experiments, see Appendix B. We also conduct preliminary tests using GTrXL and Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) on the finite environments without extensive hyperparameter tuning. GTrXL is originally based on the DRL algorithm V-MPO (Song et al., 2020). We modify our TrXL baseline to adopt the gating mechanism. Results for GTrXL and LSTM on the finite environments are provided in Appendix D. These models usually underperformed when applied directly, suggesting that out-of-the-box applications might necessitate hyperparameter optimization.

Additionally, we reaffirm that Memory Gym's environments require memory by training naive baselines, such as frame stacking (Appendix C). A selection of wall-time efficiency statistics is available in Appendix E.

5.2 Evaluation Protocol

A major challenge in DRL research is the statistical uncertainty resulting from high computational costs and the stochastic nature of training algorithms (Agarwal et al., 2021; Jordan et al., 2024). Due to these expenses, many studies rely on limited samples, such as those evaluating PPO, MRA, and HCAM with only three runs (i.e., seeds) per experiment. While our empirical protocol does not fully resolve this issue, it mitigates it by repeating each experiment five times on independent seeds. The code and data behind all plots are publicly available at `https://marcometer.github.io/jmlr_2024.github.io/`.

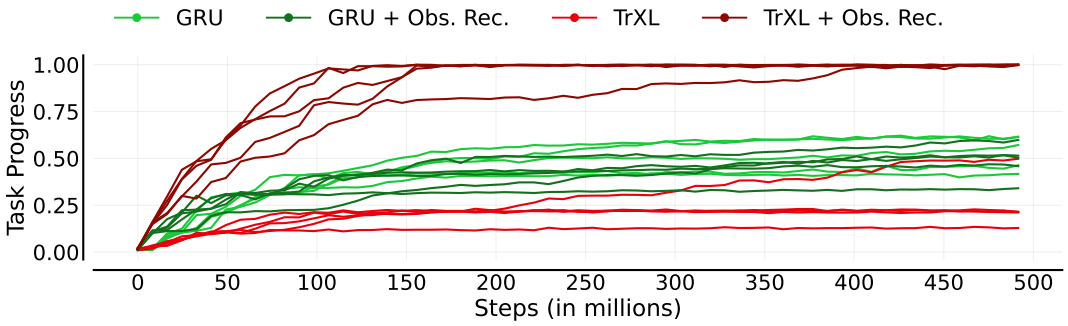
The forthcoming results on the finite and endless environments (Sections 5.3 and 5.4) display the mean performance of each individual run. Instead of aggregating these runs, we take advantage of the available space to provide denser information. Each data point of a single run is an average of 50 episode seeds, repeated three times to account for the agent's stochastic policy. Although five samples may still be insufficient for drawing strong conclusions, we increase reliability by aggregating the trained agents' performances across both finite and endless environments. These aggregation plots (Figures 6(d) and 7(d)) illustrate the mean and standard deviation of 15 independent seeds, allowing for a more robust statistical analysis. The remaining sample efficiency curves, which are used for supplementary experiments (e.g., exploring TrXL's low effectiveness on endless tasks), average five runs and display standard deviations.

5.3 Finite Environments: Transformer-XL Relies on Obs. Reconstruction

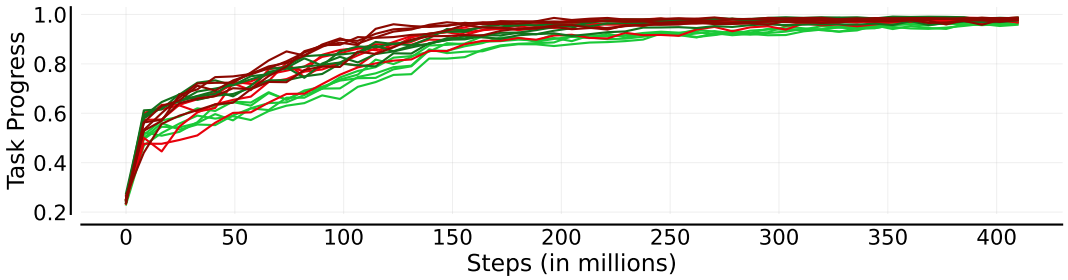
The introduction of the TrXL baseline motivates conducting benchmarks on the finite environments of Memory Gym, as depicted in Figure 6. Instead of comparing the success rates on the full tasks, we consider a more fine-grained task progression that provides denser information.

In Mortar Mayhem (Figure 6(a)), the task is completed when an agent successfully executes 10 commands. The only agent that completes the entire task is the TrXL agent, which benefits from observation reconstruction. The GRU agent, which does not use this auxiliary loss, achieves an average completion of 6.2 commands at best and 4.2 commands at worst. TrXL without reconstruction ranges from 1.3 to 5 commands.

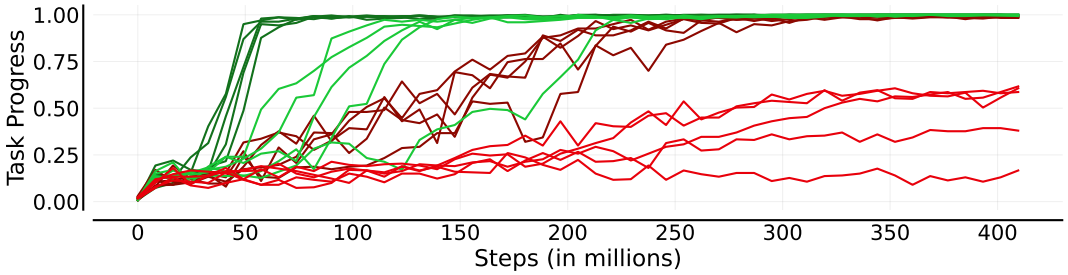
In Mystery Path, task progress is based on the proportion of the path completed. The path lengths of the evaluated episodes vary between 7 and 14 tiles. In this environment,



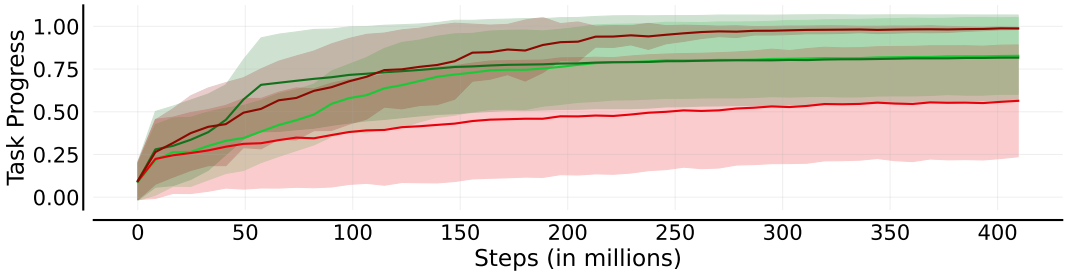
(a) Mortar Mayhem



(b) Mystery Path



(c) Searing Spotlights



(d) Aggregation and Normalization Across Finite Tasks

Figure 6: Performances across Memory Gym's finite environments. For each agent, the mean performance of 5 independent seeds is shown (a, b, and c). (d) shows the mean performance and the standard deviation when normalizing and aggregating across the finite tasks.

all agents are effective, with the TrXL agents being more sample efficient than their GRU counterparts. However, when wall-time efficiency is compared, TrXL and GRU are about on par because TrXL is slower (Annex E).

In Searing Spotlights (Figure 6(c)), the reported task progression measures 50% for collecting the coin, with the remainder dedicated to successfully terminating the episode by using the exit. The results demonstrate that utilizing the observation reconstruction loss is key, significantly improving the effectiveness of TrXL and the sample efficiency of GRU. This improvement is due to rare events where the agent, the coin, and the exit are seldomly visible. Thus, the visual encoder benefits from the auxiliary learning signal provided by the observation reconstruction. Nevertheless, both GRU variants, exhibit notably higher sample efficiency compared to TrXL.

As a final assessment, we aggregate and average the normalized task progression in Figure 6(d). Each curve represents the mean and standard deviation of 15 independent runs. This statistical measure demonstrates that TrXL relies on observation reconstruction to complete 99% of the tasks on average. Without it, its mean task progression drops to 56%. GRU without reconstruction loss reaches a task progression of 83%. When leveraging observation reconstruction, GRU appears more sample efficient, with a mean task completion of 82%.

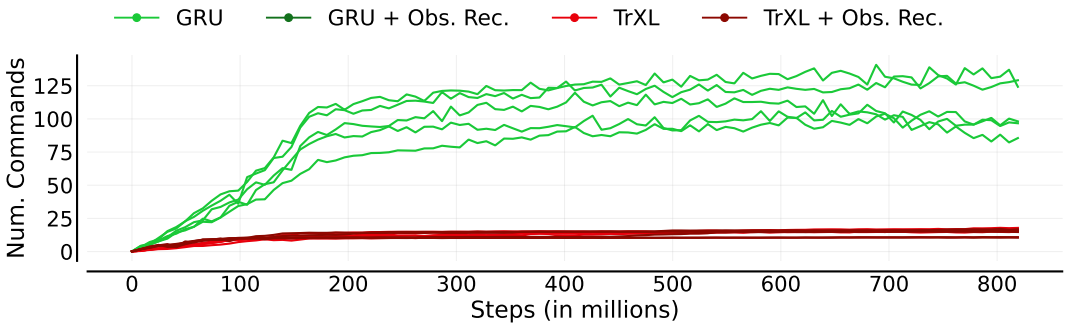
5.4 Endless Environments: GRU is more effective than Transformer-XL

To our surprise, across all three endless environments, the recurrent agent consistently proves to be much more effective than the transformer-based agent. This diverges from the observations made by Parisotto et al. (2020) and Lampinen et al. (2021), where an LSTM was found to be less effective. This subsection presents the raw results, and the following one discusses how finite tasks provide only a truncated view of the memory capabilities of the evaluated agents.

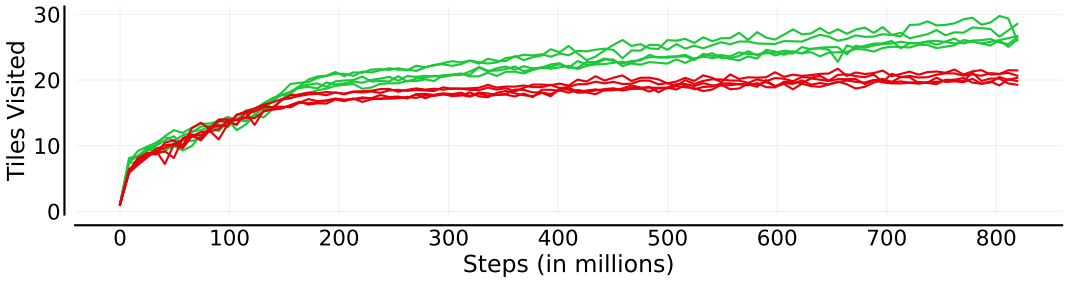
Notably, the most substantial gap between GRU and TrXL emerges in the results of Endless Mortar Mayhem (Figure 7(a)). In this context, GRU attains an impressive range of 84 to 120 executed commands using roughly 300 million steps, whereas both TrXL baselines only attain an average of 17 commands. The best data point for the recurrent agent measures 140 successfully executed commands on average for a single training run. When taking the mean of the 5 GRU runs, the best point in time scores 115 commands with a standard deviation of 15.

The outcomes depicted in Figure 7(b) illustrate the results obtained from the Endless Mystery Path environment. In this case, GRU remains more effective than TrXL, although the gap between their performances is narrower compared to Endless Mortar Mayhem. In this setting, the best TrXL run visits 21 tiles of the path, while GRU visits 29 tiles at best. After roughly 150 million steps, all GRU curves continuously distance themselves from TrXL. Upon training completion, the mean of all GRU runs is 26 tiles, while the TrXL ones average 20 tiles.

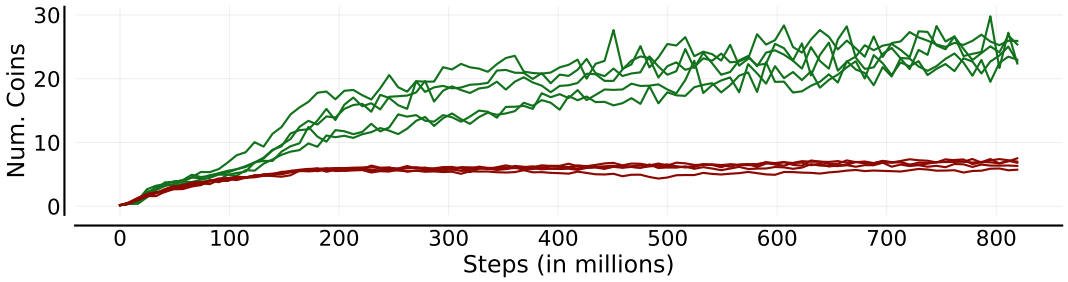
In Endless Searing Spotlights (Figure 7(c)), both agents utilize the observation reconstruction loss. TrXL is capable of collecting 7.5 coins at maximum, while GRU's best performance is nearly 4 times higher with 29.8 collected coins. The final performance after training measures 23.9 coins for GRU and 6.7 ones for TrXL.



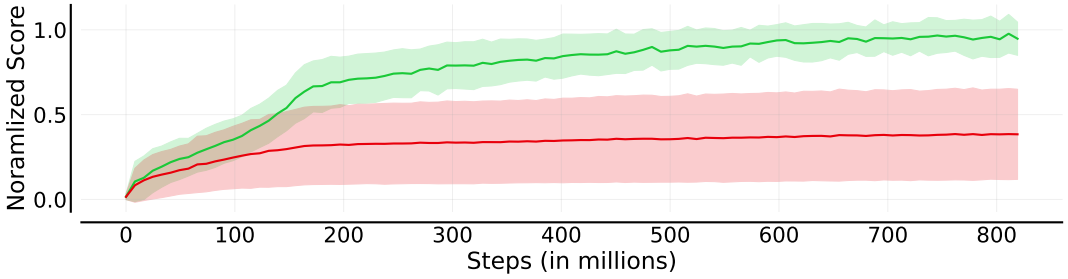
(a) Mortar Mayhem



(b) Mystery Path



(c) Searing Spotlights



(d) Aggregation and Normalization Across Finite Tasks

Figure 7: GRU's consistent superiority over TrXL is revealed in Memory Gym's endless environments. For each agent, the mean performance of 5 independent seeds is shown (a, b, and c). (d) shows the mean performance and the standard deviation when normalizing and aggregating across the endless tasks.

All results obtained from the endless environments show a clear performance gap between GRU and TrXL. This is also salient in the aggregation plot of Figure 7(d), which combines all GRU and TrXL runs, including those from Endless Searing Spotlights that use observation reconstruction. Hence, both the green and red curves aggregate 15 independent seeds. The data is normalized by the maximum mean score of the GRU agent in each environment. Figure 7(d) shows that the GRU agent peaks at 0.98, while TrXL reaches a normalized score of 0.39. Additionally, the error bars of the standard deviations do not overlap after roughly 250 million steps.

5.5 Memory Effectiveness: Finite Boundaries vs. Endless Potential

In this subsection, we highlight that the insights derived from finite environments provide only a truncated view of the true memory capabilities of the benchmarked agents. To recall, higher memory effectiveness implies the ability to memorize more information over extended periods.

Mortar Mayhem stands as an exception among finite environments, where varied levels of effectiveness are observable. One might assume that these differences would align with the results from endless tasks, yet this alignment does not occur. Notably, in the endless setting, GRU completes significantly more commands than the 6.2 observed in the finite environment. This increased quantity may be attributed to two key factors. Firstly, the environment's curriculum begins at an easier level by only demanding one command initially. Secondly, Endless Mortar Mayhem allows for two strategies for memorizing commands: the direct visualization of the command and the agent's ability to infer commands based on its past actions. This may allow for the agent's policy to adapt faster to growing difficulties. Regarding episode lengths, when completing 10 commands in the finite tasks, episodes last for 274 steps. In the endless task, GRU plays for more than 3000 steps and TrXL for more than 400 steps.

In the Mystery Path environment, challenges are limited to a maximum path length of 14 tiles, providing only a basic indication of the memory requirements. This measure is approximate, as the procedurally generated path may include multiple turns, increasing the likelihood of the agent falling off the path. Both GRU and TrXL effectively master this challenge, but they surpass these capabilities in the endless version. While successful episodes in the finite task last for about 100 steps, GRU reaches more than 600 steps on average in the endless task, while TrXL approaches 400 steps.

Quantifying performance in Searing Spotlights involves navigating to two entities (coin and exit), while the agent must maintain its hidden position. In this finite task, effective episodes typically last fewer than 50 steps. However, in Endless Searing Spotlights, GRU and TrXL can gather more coins and maintain their hidden positions for 691 and 240 steps on average, respectively. This performance demonstrates that both agents not only achieve higher scores but also sustain their concealed positions for longer durations than observed in the finite version of Searing Spotlights.

To conclude, endless tasks demonstrate their potential to reveal more effective memory capabilities in both baselines beyond what is evident in finite tasks, as more information needs to be memorized and maintained over prolonged time horizons.

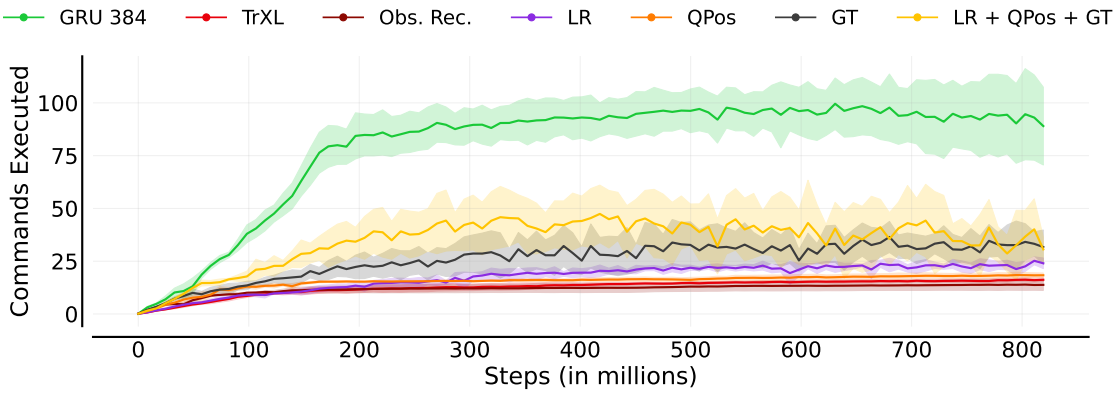


Figure 8: Experiments on Endless Mortar Mayhem with varied TrXL configurations. The learning rate (LR) is adjusted to decay from 2.75e-4 to 1.0e-4 over 160 million steps, compared to the previous range from 2.75e-4 to 1.0e-5. We also incorporate observation reconstruction (Obs. Rec.) and observe the greatest improvement based on a ground truth estimation head (GT) as sanity check. Another test augments TrXL's query with absolute positional encoding (QPos). The final experiment combines the optimized learning rate, augmented query, and ground truth estimation.

To conclude, finite environments provide only a truncated view of the agents' capabilities. Instead, the endless environments demonstrate their potential to reveal more effective memory capabilities in both baselines beyond what is evident in finite tasks, as more information needs to be memorized and maintained over ever-growing time horizons.

5.6 Investigating Transformer-XL's Surprisingly Low Effectiveness

The unexpectedly low performance of TrXL prompts us to examine several hypotheses. Endless Mortar Mayhem, presenting the largest performance gap, serves as our primary environment for these investigations.

5.6.1 Inadequate Networks Capacity

Our TrXL baseline comprises 2.8 million trainable parameters, while GRU consists of 4.05 million, prompting the question of whether TrXL's model architecture lacks the necessary capacity. To investigate this, we conducted experiments varying the number of layers (2, 3, 4), the embedding dimension size (256, 384, 512), and the memory window length (256, 384, 512). None of these adjustments led to closing the performance gap to GRU. Even when the GRU cell is scaled down to a dimension of 384, totaling 2.7 million parameters, it still surpasses TrXL by a large margin (Figure 8). Note that TrXL's maximum context length, defined by the number of layers and window length (Equation 7), reaches 766 timesteps with 3 layers and a window length of 256. Therefore, the context length is not the limiting factor in any endless environment.

Scaling up TrXL also increases its demand for GPU memory. When scaling up multiple architecture details, our implementation and hyperparameters may cause the training to exceed the available 40GB GPU memory of an NVIDIA A100. Workarounds include reducing the batch size or transferring training data between the GPU and CPU, but these options significantly worsen wall-time efficiency. Furthermore, another indication of untapped capacity lies in the amplification of the learning signal, which we elaborate on in the following section.

5.6.2 Weak Learning Signal

Figure 8 depicts several additional experiments aimed at amplifying the learning signal. We made a naive adjustment to the learning rate schedule, utilized observation reconstruction, and introduced a ground truth estimation head. This ground truth estimation head is added to the actor-critic model architecture and predicts the next target position to move to. Labels are provided by the environment, and the estimates are optimized using the mean-squared error loss. While ground truth information is typically unavailable and considered inappropriate for this benchmark, it serves as a useful sanity check, helping to determine if an additional learning signal is advantageous in this context.

When training incorporates ground truth estimation, there is improvement in the agent's policy. Previously, the mean number of completed commands stood at 16; with ground truth estimation, it reaches a mean of up to 36 commands. As a default and during the initial 160 million training steps, the learning rate linearly decays from $2.75 \times 1.0\text{e-}4$ to 1.0e-5. By setting the final decayed learning rate to 1.0e-4, the mean performance hits 25 commands. The results of leveraging ground truth estimation or tuning the learning rate schedule imply that the model's current capacity is capable of retaining more information.

5.6.3 Lack of Temporal Information in the Initial Query

The next hypothesis pertains to the initial query of the first TrXL layer. This query is constructed solely based on the features extracted from the observation encoders at the current time step, encompassing only minimal temporal information. In contrast, the query of the second layer draws from the aggregated outcome of the memory window, thus capturing more substantial temporal information. However, we believe that the initial query could be further enriched with additional information. In Figure 8, one experiment augments the query with absolute positional encoding, providing direct access to the current time step. Despite this enhancement, the agent's performance only moderately improves, reaching a mean of 18 completed commands. We also explored embedding the original query with the previous output of the TrXL block. However, this simple approach did not yield positive results.

None of the aforementioned measures individually reach the performance level of the GRU agent. However, when combined, they achieve a mean performance of 47 executed commands at best.

5.6.4 Harmful Off-Policy Data in the Episodic Memory

Next, it can be discussed whether the combination of PPO and the cached hidden states of TrXL's episodic memory pose a threat. PPO is an on-policy algorithm that consumes

the training data for a few epochs. Once the second training epoch on the same batch commences, the batch is already considered off-policy. However, PPO's loss function can mitigate this issue and hence allows for several epochs. But this mitigation is likely limited and we wonder whether this issue is enlarged by the cached hidden states. In our current training setup, the agent collects 512 steps of data. If an episode exceeds 512 steps and is still running, the older hidden states from that episode become outdated over multiple iterations of PPO. To address this, future research should investigate the impact of stale hidden states on training efficacy or how one can solve the issue of truncated episodes during optimization. One potential strategy could involve computing more fresh hidden states alongside utilizing cached ones. Alternatively, cached hidden states could be periodically recomputed, albeit at the expense of increased inference overhead. Drawing parallels, Kapturowski et al. (2019) explored similar challenges with their recurrent, off-policy algorithm R2D2.

5.6.5 Indistinguishable Absolute Positional Encoding

The final hypothesis concerns a potential limitation of the absolute positional encoding. Without positional encoding, the agent cannot differentiate the time between past observations. As a default, we rely on absolute positional encodings. This encoding always spans across the potential maximum episode length. In the finite environments, the longest range is set to 512, being the sequence length, which is also used in the original transformer (Vaswani et al., 2017). For the endless tasks, this encoding is set up for 2048 steps, an episode duration that is yet unreachable to the TrXL agents.

The original work of TrXL introduces a relative positional encoding (Dai et al., 2019) as a consequence of extended context lengths. However, when they refer to the absolute positional encoding, they only apply it to the range of the current window. If a window has a length of 4, the positions within the window are always the same (i.e. $[0, 1, 2, 3]$), even though the window moves across the entire sequence. In our case, the absolute positional encoding considers the entire sequence. If the window has progressed in time, the applied positional encoding relies on the current time step (i.e. $[t-3, t-2, t-1, t]$).

Nevertheless, the absolute positional encoding needs to be defined prior training. When scaling it up, the question arises whether the agent's model is capable of differentiating time steps, because a longer length indicates a more fine-grained sample frequency over the underlying positional encoding. To test this, we trained a TrXL agent on the finite task of Mortar Mayhem and configured an absolute positional encoding of length 1024 and 2048. Both experiments, which were repeated twice, resulted in an agent that poorly completes only 1 to 2 commands. This demands for questioning the absolute positional encoding in endless tasks. But if the elements of the absolute position encoding of length 2048 are not distinguishable, how do those agents master more than 10 commands in Endless Mortar Mayhem?

If this assumption is correct—that the absolute positional encoding is the culprit in our TrXL baseline—leveraging learned or relative positional encodings instead should improve performance. As seen in Figure 9, no notable advancement is accomplished by both variants. We observed two outliers among the 5 runs that utilized relative positional encoding. One underperformed, completing only 16 commands on average, while the other excelled,

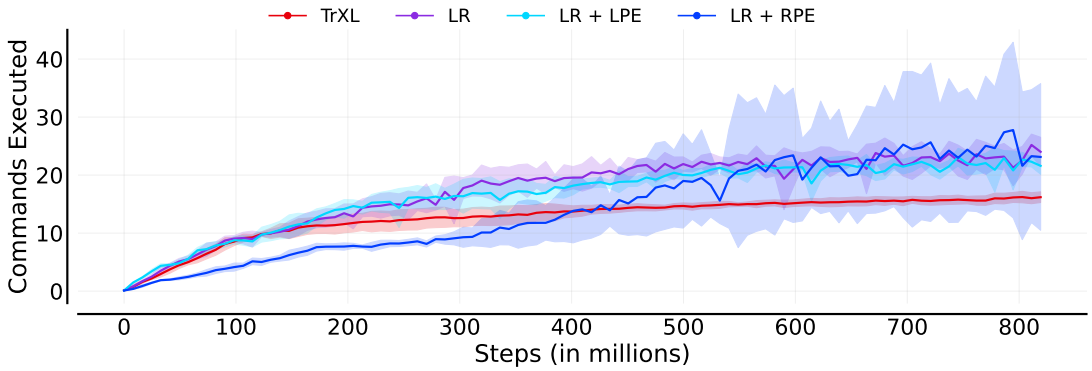


Figure 9: Experiments on Endless Mortar Mayhem with varied positional encoding. The learning rate schedule (LR) is adjusted to decay from 2.75e-4 to 1.0e-4 over 160 million steps, compared to the previous 1.0e-5. The acronym LPE refers to a learned positional encoding, while RPE depicts the relative one.

finishing 56 commands on average. Given this discussion, a more in-depth exploration of positional encoding in this context is advisable.

5.7 Recurrence is not Vulnerable to Spotlight Perturbations

Training recurrent agents in Searing Spotlights with previously established hyperparameters (refer to the last column of Table 4) does not yield a meaningful policy. Despite numerous attempts at hyperparameter tuning, architecture adjustments, and task modifications, Searing Spotlights remained as a puzzle. Even with full observability, where all entities are permanently visible, the performance is catastrophic. These challenges prompted the hypothesis that spotlights might introduce noise, heavily affecting the agent's policy (Pleines et al., 2023). Support for this theory came from experiments in a simplified BossFight environment (Cobbe et al., 2020), using identical spotlight dynamics as in Searing Spotlights, suggesting a potential vulnerability of recurrent agents to such perturbation (Pleines et al., 2023).

However, the results presented in Figure 6(c) neglect this hypothesis. This prompted us to conduct an ablation study, beginning with the previous hyperparameters. In this study, only a single hyperparameter or environment option was replaced with the current one at a time. Table 1 presents the results for Searing Spotlights when full observability is ensured. Initially, observation reconstruction was presumed to be the solution. However, the crucial adjustment proved to be the decision to avoid normalizing advantage estimates during optimization. Only the GRU agent, which omits advantage normalization, and the memory-less agent, based entirely on previous hyperparameters, recorded noteworthy success rates of 100% and 96%, respectively.

A salient observation emerges when tracking the gradient norms across the agent models during training. With normalized advantages, the gradient norm is typically more than 12 times greater than when normalization is bypassed. This suggests that during gradient

Table 1: Results from the ablation study on Searing Spotlights with full observability. Each experiment was run 5 times, though the data sourced directly from the training process deviates from our standard protocol. Every PPO iteration (of 5,000 total) produces a data point that reflects the average over the last 100 episodes. Rows are ordered in descending fashion by the mean success rate. The mean is computed from the top-performing data point, averaged across repetitions, from the specific experiment. Consequently, the time point is represented as a percentage duration. The gradient norm is accumulated over all data points within the given duration.

Experiment	**Value**		**Success Rate**		**Duration**	**Gradient Norm**	
	New	Old	Mean	Std		Mean	Std
Advantage Norm.	Off	On	1.00	0.00	31%	0.02	0.01
No Memory			0.96	0.11	99%	0.28	0.03
Agent Health Points	5	10	0.79	0.35	99%	0.30	0.02
Transformer-XL			0.69	0.27	99%	0.30	0.03
Agent Speed	3.0	2.5	0.58	0.22	65%	0.33	0.06
Clip Range	0.1	0.2	0.52	0.25	99%	0.27	0.05
Max Episode Length	256	512	0.48	0.27	99%	0.37	0.04
Old Hyperparameters			0.43	0.06	96%	0.38	0.06
Observation Rec.	On	Off	0.39	0.06	96%	0.40	0.04
Max Gradient Norm	0.25	0.5	0.36	0.05	99%	0.24	0.01

descent, excessively large optimization steps might be taken, resulting in an oscillating behavior that fails to converge to a local minimum. Simultaneously, unnormalized advantages remain at a scale of 2e-4, but post-normalization, the magnitudes stretch beyond 5. Considering that the policy ratio is modulated by these advantage estimates (Schulman et al., 2017), normalization introduces a strikingly different scale than its absence. Standard practice typically normalizes advantages by subtracting the mean and dividing by the standard deviation, often applied at the mini-batch level.

Several comprehensive studies have delved into such nuances of PPO's implementation (Engstrom et al., 2020; Andrychowicz et al., 2021; Huang et al., 2022a). Among them, only Andrychowicz et al. (2021) turned off advantage normalization and found negligible performance impact. However, their extensive study focused on control tasks with undiscounted returns in the thousands, contrasting Memory Gym's finite environments where the undiscounted returns do not exceed 2. During hyperparameter optimization, turning off advantage normalization also enhanced performance in Mortar Mayhem and Mystery Path. Given the critical importance of utilizing raw advantages, there remains an opportunity to delve even deeper into PPO to comprehend and refine its essential aspects.

6. Conclusion

In this study, we advanced Memory Gym's environments to novel endless tasks, tailored to benchmark memory-based Deep Reinforcement Learning algorithms. These tasks feature

a mounting challenge similar to the cumulative memory game "I packed my bag". As the agent's policy refines, the task dynamically expands, serving as an automatic curriculum. This innovative framework enables a thorough evaluation of memory-based agents, emphasizing their overall effectiveness beyond mere interaction efficiency with the environment. Our experimental results reveal that agents trained on these endless tasks consistently show stronger capabilities than those trained on finite tasks, demonstrating that finite environments offer only a truncated view on an agent's full memory capabilities.

To foster a more inclusive research landscape, we contributed an open-source PPO baseline powered by Transformer-XL (TrXL). This baseline employs an attention mechanism applied to an episodic memory with a sliding window. When benchmarked against Memory Gym, the TrXL baseline competes head-to-head with another prominent baseline found in Gated Recurrent Unit (GRU). Notably, in finite environments, TrXL is more effective than GRU, but strongly depends on the observation reconstruction loss. Our most unexpected revelation is the comeback of GRU in the endless environments. GRU surpasses TrXL by large margins, while also being computationally more efficient. Further probing into the Endless Mortar Mayhem environment revealed only marginal improvements for TrXL upon enriching the learning signal.

This prompts future investigations into potential reasons behind TrXL's limitations, such as the possibility of episodic memory staleness or an initial query lacking temporal awareness. As we move forward, it will be compelling to discern the performance thresholds of other memory mechanisms and DRL algorithms. In addition to examining more attention-based and recurrent architectures, it may be worth exploring structured state space sequence models (Gu et al., 2022), given their recent debut in DRL (Lu et al., 2023).

Acknowledgements

Our sincere appreciation goes to Vincent-Pierre Berges and Fabian Ostermann for their enlightening discussions that greatly enriched our work. We also extend our gratitude to Andrew Lampinen for his invaluable insights on utilizing Transformer-XL as episodic memory. Special thanks are due to Günter Rudolph for his unwavering support. This project would not have been possible without the generous computing time provided by the Paderborn Center for Parallel Computing (PC2) and the support from the Linux-HPC-Cluster (LiDO3) at TU Dortmund. We are deeply thankful for all of their contributions.

Appendix A. Environment Parameters

Table 2: Default reset parameters of the finite environments. Parameters marked with an asterisk (*) indicate uniform sampling. Values enclosed in square brackets represent discrete choices, while values in parentheses denote a range from which sampling is performed. For Mortar Mayhem Grid and Mystery Path Grid, the parameters remain the same as its parent, with only the modified ones presented.

Mortar Mayhem	
Parameter	**Default**
Agent Scale	0.25
Agent Speed	3
Arena Size	5
No. Available Commands	9
No. Commands*	[10]
Command Show Duration*	[3]
Command Show Delay*	[1]
Execution Duration*	[6]
Execution Delay*	[18]
Show Visual Feedback	True
Reward Command Failure	0
Reward Command Success	0.1
Reward Episode Success	0
Mortar Mayhem Grid	
Parameter	**Default**
No. Available Commands	5
Execution Duration*	[2]
Execution Delay*	[6]
Mystery Path	
Parameter	**Default**
Max Episode Length	512
Agent Scale	0.25
Agent Speed	3
Cardinal Origin Choice*	[0, 1, 2, 3]
Show Origin	False
Show Goal	False
Show Visual Feedback	True
Reward Goal	1
Reward Fall Off	0
Reward Path Progress	0.1
Reward Step	0
Mystery Path Grid	
Parameter	**Default**
Max Episode Length	128
Reward Path Progress	0

Searing Spotlights	
Parameter	**Default**
Max Episode Length	256
Agent Scale	0.25
Agent Speed	3
Agent Always Visible	False
Agent Health	5
Sample Agent Position	True
Use Exit	True
Exit Scale	0.5
Exit Visible	False
Number of Coins*	[1]
Coin Scale	0.375
Coin Always Visible	False
No. Initial Spotlight Spawns	4
No. Spotlight Spawns	30
Spotlight Spawn Interval	30
Spotlight Spawn Decay	0.95
Spotlight Spawn Threshold	10
Spotlight Radius*	(7.5-13.75)
Spotlight Speed*	(0.0025-0.0075)
Spotlight Damage	1
Light Dim Off Duration	6
Light Threshold	255
Show Visual Feedback	True
Show Last Action	True
Show Last Positive Reward	True
Render Background Black	False
Hide Checkered Background	False
Reward Inside Spotlight	0
Reward Outside Spotlights	0
Reward Death	0
Reward Exit	1
Reward Coin	0.25
Reward Max Steps	0

Table 3: Default reset parameters of the endless environments. Parameters marked with an asterisk (*) indicate uniform sampling. Values enclosed in square brackets represent discrete choices, while values in parentheses denote a range from which sampling is performed. If the max episode length is equal or less than zero, episodes will not terminate because of reaching the max episode length.

Endless Mortar Mayhem	
Parameter	**Default**
Max Episode Length	-1
Agent Scale	0.25
Agent Speed	3
No. Available Commands	9
Command Show Duration*	[3]
Command Show Delay*	[1]
Execution Duration*	[6]
Execution Delay*	[18]
Show Visual Feedback	True
Reward Command Failure	0
Reward Command Success	0.1
Endless Mystery Path	
Parameter	**Default**
Max Episode Length	-1
Agent Scale	0.25
Agent Speed	3
Show Origin	False
Show Past Path	True
Show Background	False
Show Stamina	False
Show Visual Feedback	True
Camera Offset Scale	5
Stamina Level	20
Reward Fall Off	0
Reward Path Progress	0.1
Reward Step	0

Endless Searing Spotlights	
Parameter	**Default**
Max Episode Length	-1
Agent Scale	0.25
Agent Speed	3
Agent Always Visible	False
Agent Health	10
Sample Agent Position	True
Coin Scale	0.375
Coin Show Duration	6
Coin Always Visible	False
Steps per Coin	160
No. Initial Spotlight Spawns	3
Spotlight Spawn Interval	50
Spotlight Radius*	(7.5-13.75)
Spotlight Speed*	(0.0025-0.0075)
Spotlight Damage	1
Light Dim Off Duration	6
Light Threshold	255
Show Visual Feedback	True
Render Background Black	False
Hide Checkered Background	False
Show Last Action	True
Show Last Positive Reward	True
Reward Inside Spotlight	0
Reward Outside Spotlights	0
Reward Death	0
Reward Coin	0.25

Appendix B. Hyperparameters

Table 4 presents the hyperparameters utilized in our final experiments unless otherwise specified. The learning rate and entropy coefficient undergo linear decay from their initial to final values. This decay occurs exclusively during the first 10,000 PPO updates (equivalent to 163,840,000 steps). As training progresses beyond this point, the final hyperparameter serves as a lower threshold. Regarding the sequence length and memory window length, they are determined based on the longest possible episode in the finite environments. In the case of endless environments, the so far best sequence length is fixed at 512 for GRU and 256 for TrXL.

Our hyperparameter search was conducted using a customized implementation built upon the optuna tuning framework (Akiba et al., 2019). The objective was to discover a

Table 4: Hyperparameters and architectural details used in our final experiments. The "Final" column denotes the selected hyperparameter values, while the "Search Space" column represents additional discrete choices explored during the tuning process. The last column details the old hyperparameters of our previous study (Pleines et al., 2023). For this column alone, empty cells indicate that the values align with those in the "Final" column. T-Fixup is a transformer weight initialization approach by Huang et al. (2020).

Hyperparameter	**Final**	**Search Space**			**Old**
Training Seeds	100000				
Number of Workers	32				
Worker Steps	512				
Batch Size	16384				
Disocunt Factor Gamma	0.995				0.99
GAE Lamda	0.95				
Optimizer	AdamW				
Epochs	3	2	4		
Number of Mini Batches	8	4			
Advantage Normalization	No	Batch	Mini Batch		Mini Batch
Clip Range Epsilon	0.1	0.2	0.3		0.2
Value Loss Coefficient	0.5	0.25			0.25
Initial Learning Rate	2.75e-4	2.0e-4	3.0e-4	3.5e-4	3.0e-4
Final Learning Rate	1.0e-5				1.0e-4
Initial Entropy Coef.	1.0e-4	1.0e-3	1.0e-2		
Final Entropy Coef.	1.0e-6				1.0e-5
Reconstruction Loss Coef.	0.1	0.5	1.0		n/a
Maximum Gradient Norm	0.25	0.35	0.5	1.0	0.5
Recurrent Neural Network					
Number of Recurrent Layers	1	2			
Embedding Layer	Yes				No
Layer Type	GRU	LSTM			
Residual	False	True			
Sequence Length	512 or max				
Hidden State Size	512	256	384		
Transformer-XL					
Number of TrXL Layers	3	2	4		
TrXL Block Dimension	384	256	512		
Block Weight Initialization	Xavier	Orthogonal	Kaiming	T-Fixup	
Positional Encoding	Absolute	None	Learned		
Number of Attention Heads	4	8			
Memory Window Length	256 or max				

set of hyperparameters that demonstrate strong performance across Memory Gym's environments, for the GRU and TrXL baselines individually. Considering limited resources and the computational expense of tuning, we restricted the search space to a limited number of

options. The majority of tuning experiments focused on the finite environments. Therefore, we do not claim to provide optimal hyperparameters.

For each environment, we established individual tuning experiments corresponding to the choices presented in Table 4. Each experiment, or trial, involved a baseline set of hyperparameters that was modified by selecting a single parameter choice from the entire search space. Notably, we did not train permutations of all available choices at this stage. To optimize resource utilization, we implemented optuna's percentile pruner, which retained the top 25 percentile of trials. Trials that performed below this percentile were pruned, but only after surpassing 2,000 PPO updates as a warm-up phase.

After completing the single experiments, we expanded the hyperparameter search by allowing for permutations sampled using optuna's Tree-structured Parzen Estimator. It is important to note that individual trials were not repeated, but since tuning was conducted across multiple environments, one could consider them as repetitions in a broader sense.

The batch size is determined by multiplying the number of environment workers by the number of worker steps performed by each worker to gather training data. To efficiently utilize the resources of a system with 32 CPU cores and 40GB VRAM on an nVidia A100 GPU, we chose a batch size of 16,384 samples. However, when scaling up the model, it is possible to exceed the available GPU memory. In such cases, there are two options: reducing the batch size or outsourcing the data to CPU memory. Both of these workarounds come with increased expenses in terms of wall-time.

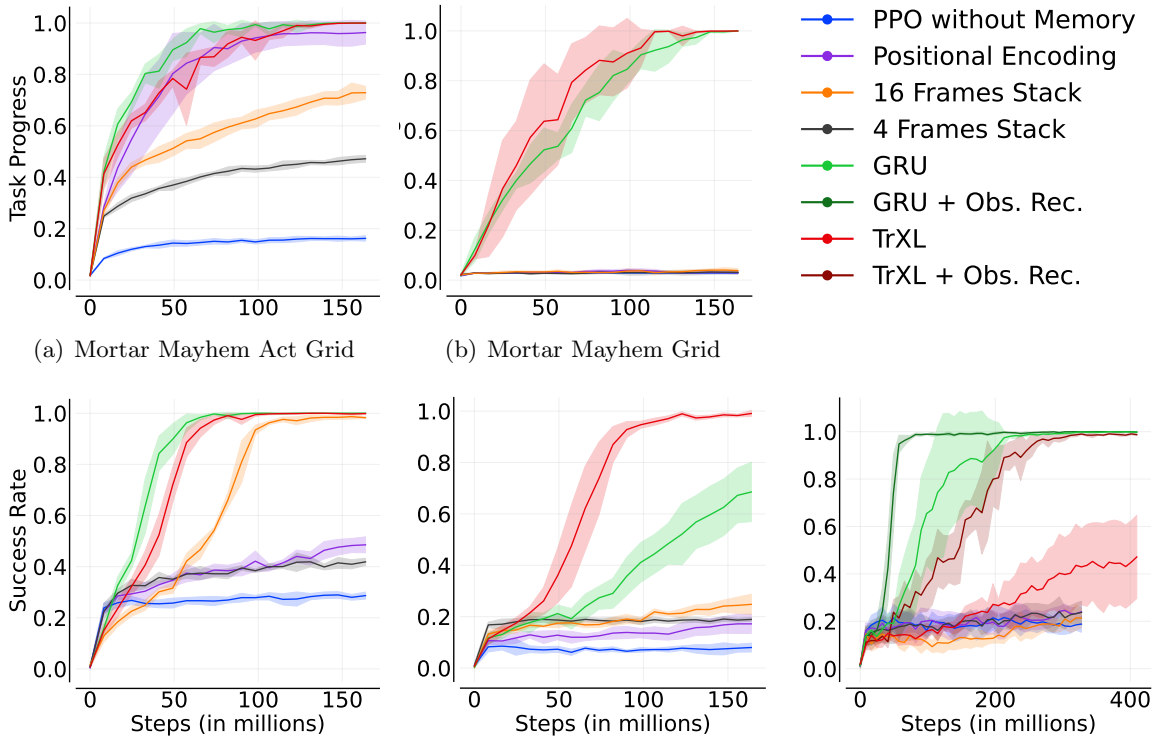
Appendix C. Results of Minor Baselines on the Finite Environments

This set of experiments reaffirms that agents trained on Memory Gym require memory. To show this, we train several naive baselines on simplified versions of the environments. These baselines include PPO without memory, PPO with frame stacking (4 and 16 grayscale frames), and PPO with absolute positional encoding (Vaswani et al., 2017) provided as vector observation. Note that the hyperparameters for these baselines are not tuned. Figure 10 shows the mean and standard deviation of 5 independent seeds for each presented agent.

In Mortar Mayhem Act Grid, where the commands are completely provided as vector observation, PPO without memory is ineffective, while the frame stacking agents achieve nearly either 4.7 or 7.3 commands with a slight upward trend, while the one with absolute positional encoding follows closely, completing 9.6 commands. The positional encoding baseline adds the episode's current time step to the agent's observation allowing for temporal awareness within the sequential execution of commands. Once the complete task is present (Figure 10(b)), only GRU and TrXL prove to be effective.

We obtain a similar impression when training on Mystery Path Grid where the origin and goal are not perceivable by the agent (Figure 10(e)). If the origin and the goal are visible, a horizon of 16 frames is sufficient to train an effective policy as shown by the frame stacking agent (Figure 10(d)). Stacking 4 frames or leveraging positional encoding leads to a success rate of 42% and 49% respectively. The memory-less agent's rate goes up to about 29%.

Limited success is proven by the minor baselines in Searing Spotlights as seen in Figure 10(f). Only in this experiment, the naive baselines also leverage observation reconstruction.

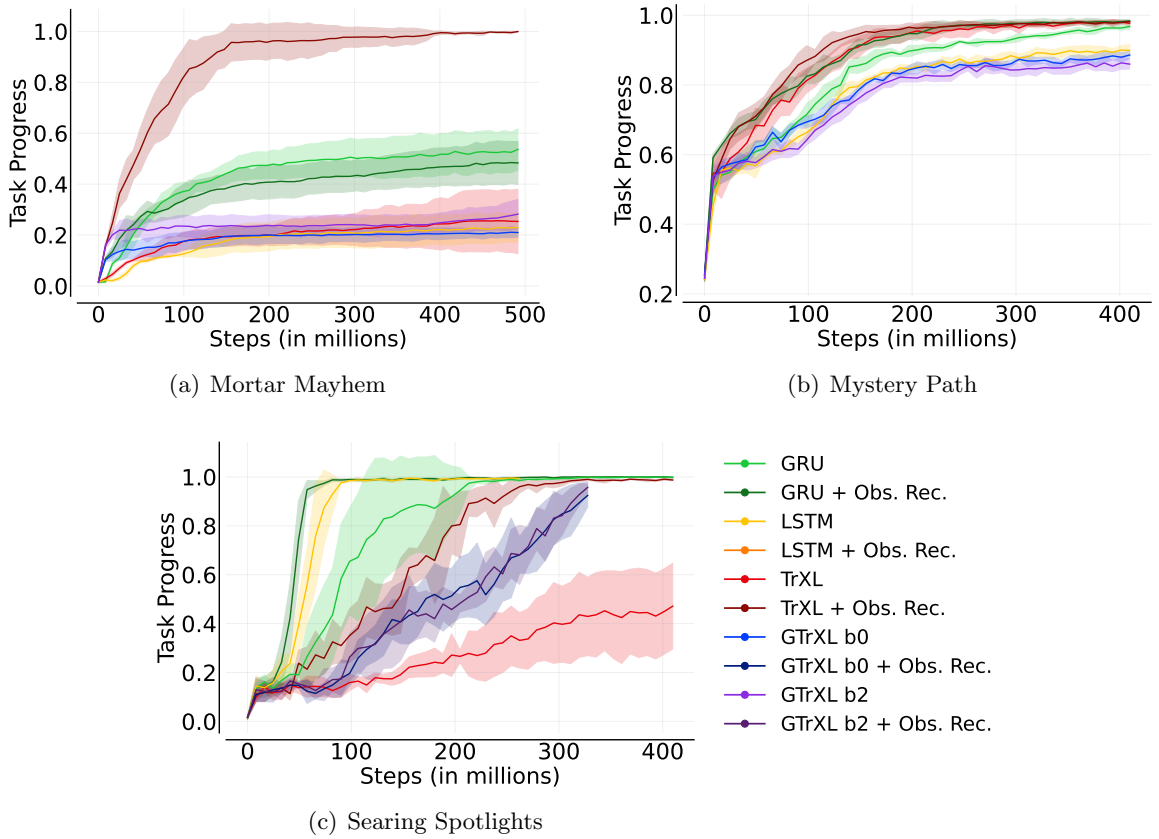


(a) Mortar Mayhem Act Grid (b) Mortar Mayhem Grid

(d) Mystery Path Grid (Cues On) (e) Mystery Path Grid (f) Searing Spotlights

Figure 10: Performance comparison of several minor baselines on instances of Memory Gym's finite environments. Mortar Mayhem Act Grid (a) provides commands as a fully observable vector, bypassing the Clue Task. Both (a) and Mortar Mayhem Grid (b) utilize grid-like locomotion. Mystery Path Grid operates on grid-like locomotion, with (d) rendering the the goal and origin to the agent's observation. (e) hides these. Searing Spotlights (f) is not varied, while the minor baselines leverage observation reconstruction (Obs. Rec.) in this environment.

Appendix D. GTrXL and LSTM Results on the Finite Environments



(a) Mortar Mayhem

(b) Mystery Path

(c) Searing Spotlights

Figure 11: GTrXL and LSTM results on the finite environments.

Figure 11 presents extended results for the finite environments, showing the mean and standard deviation across 5 independent runs for each trained agent. The performance metrics for TrXL and GRU agents remain consistent with those depicted in Figure 6. We include results from agents utilizing Gated Transformer-XL (GTrXL) (Parisotto et al., 2020) and the Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997). These models are benchmarked without any hyperparameter tuning. For GTrXL, biases of both 0 and 2 are tested, based on the suggestion by Parisotto et al. (2020) that a larger bias might enhance learning. All additional agents in the Searing Spotlights environment employ the observation reconstruction loss (Obs. Rec.), while it is omitted in the Mystery Path environment.

For Mortar Mayhem (as seen in Figure 11(a)), the LSTM agent settles at an average of 2.2 commands, whereas the GTrXL agents achieve a mean of 2.1 and 2.8 commands. In the Mystery Path results (Figure D), the LSTM agent completes 90% of the task. The GTrXL agent with a bias of 0 reaches a 88% progression, while the one with a bias of 2 achieves 86%, challenging the earlier assertion by Parisotto et al. (2020). In the Searing Spotlights environment, the LSTM agent with observation reconstruction is slightly less efficient than

its GRU counterpart. However, GTrXL agents, though still growing to success, lag behind TrXL with observation reconstruction.

We avoid making conclusive statements based on these results. This caution stems from the fact that the LSTM and GTrXL baselines are not tuned.

Appendix E. Wall-Time Efficiency

Table 5: Wall-time efficiency metrics for various agents trained on Endless Mortar Mayhem. Agents are arranged from left to right based on increasing mean training duration, measured in hours. Other values are represented in seconds and comprise the time needed for a single update (i.e. one complete iteration of running PPO). The second row indicates if the neural network was compiled before training. Note: TrXL* metrics use 4k initial training data points, while other runs utilize 150k data points covering the full training process.

	GRU	TrXL	TrXL + QPos	TrXL + Obs. Rec.	TrXL + LR + QPos + GT	TrXL*
Compiled	No	Yes	Yes	Yes	Yes	No
Mean	4.11	5.53	5.6	5.76	5.86	7.97
Std	0.35	1.12	1.31	1.1	1.68	0.25
Min	3.6	5.2	5	5.4	5.3	7.6
Median	4.1	5.4	5.5	5.7	5.8	7.9
IQM	4.03	5.48	5.53	5.72	5.82	7.93
Mean Duration	57.08	76.81	77.78	80	81.39	110.69

Reliably reporting the wall-time of all conducted experiments is not feasible due to varying circumstances as varying hardware or different loads of the file system. Therefore, we only provide a selection of measurements on Endless Mortar Mayhem and Mystery Path.

Table 5 presents wall-time efficiency metrics for various agents trained on Endless Mortar Mayhem, utilizing the noctua2 high-performance cluster[1]. We leverage 32 CPU cores (AMD Milan 7763), an NVIDIA A100 GPU, and 100GB RAM for one training run. With PyTorch version 2, models can be lazily compiled at training onset, reducing TrXL's training time by a significant 30%. Previously, plain TrXL training averaged 110 hours, but this has been reduced to 76 hours. Incorporating observation reconstruction adds roughly 3 hours, and with ground truth estimation, it extends to 81 hours. The GRU agents are the most wall-time efficient, completing in approximately 57 hours. Thus, recurrence emerges as the most effective and efficient architecture in both sample and wall-time metrics for the endless environments.

We further take a look at the wall-time of various agents trained on Mystery Path (Figure 12). It becomes apparent that TrXL and GRU are quite on par concerning wall-time efficiency, whereas TrXL needs fewer samples as shown in section 5.3. On first sight, it seems surprising that GTrXL is faster than TrXL (Table 6) even though GTrXL is more

1. https://pc2.uni-paderborn.de/de/hpc-services/available-systems/noctua2

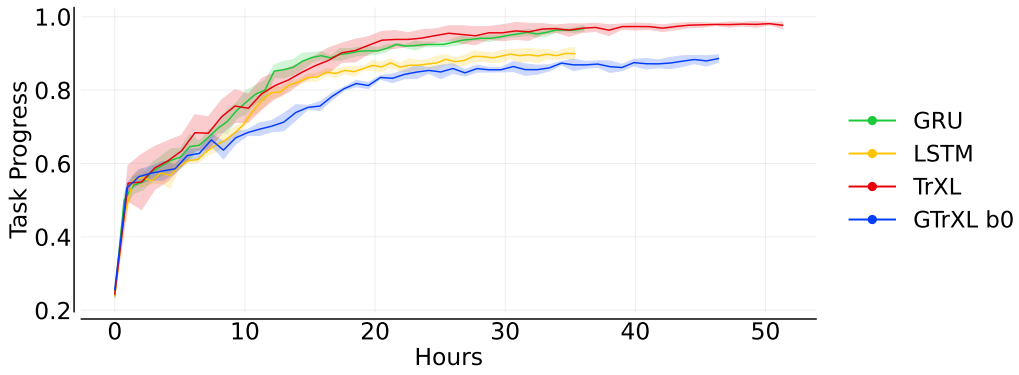


Figure 12: Wall-time efficiency curves retrieved from Mystery Path

complex. This is due to the more expensive resets of the environment. Better policies have shorter episodes and thus reset more frequently impairing wall-time. So as GTrXL's and LSTM's policies are inferior, their wall-time is faster. These results were obtained from the LiDo3 high-performance cluster[2]. Each run utilized 32 cores (AMD Epyc 7542), an A100 GPU, and 200GB RAM.

Table 6: Wall-time efficiency metrics for various agents trained on Mytsery Path. Agents are arranged from left to right based on increasing mean training duration, measured in hours. Other values are represented in seconds and comprise the time needed for a single update (i.e. one complete iteration of running PPO). The second row indicates if the neural network was compiled before training.

	LSTM	GRU	GTrXL B0	TrXL
Compiled	No	No	Yes	Yes
Mean	5.09	5.19	6.68	7.39
Std	0.22	0.28	0.31	0.24
Min	4.2	4.0	5.8	6.4
Max	7.6	7.0	62.2	11.5
Median	5.1	5.2	6.6	7.4
IQM	5.09	5.18	6.64	7.39
Mean Duration	35.35	36.04	46.4	51.32

2. https://lido.itmc.tu-dortmund.de/

Appendix F. Simulation Speed of the Environments

Memory Gym's environments are implemented using Python and PyGame[3]. The simulation speeds are shown in Table 7. At first glance, the endless variants appear slower than the finite ones. This discrepancy is attributed to the shorter episodes in endless variants, which rely on less effective constant action policies, thereby increasing the impact of environment reset costs. Compared to related memory benchmarks, our environments perform well, with only Procgen (Cobbe et al., 2020) being faster (Pleines et al., 2023). The training costs in our experiments are primarily due to the inference time of the baseline models.

Table 7: Comparison of the simulation speed of Memory Gym's environments. A constant action is executed to measure the speed, and the episode reset duration is included in the steps per second. The measurements were done on an AMD Ryzen 7 2700X, aggregating 1000 episodes per environment.

Environment	**Mean (steps/sec)**	**Std (steps/sec)**
Mystery Path	14033	158
Mortar Mayhem	13187	208
Endless Mortar Mayhem	9330	452
Endless Mystery Path	7278	555
Endless Searing Spotlights	5828	106
Searing Spotlights	5789	165

3. `https://www.pygame.org`

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Proceedings of the 34th Advances in Neural Information Processing Systems*, pages 29304–29320, 2021.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, Anchorage, Alaska, 2019.

Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters for on-policy deep actor-critic methods? a large-scale study. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

Bowen Baker, Ingmar Kanitscheider, Todor M. Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *Proceedings of the 8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2020.

Charles Beattie, Joel Z. Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, Julian Schrittwieser, Keith Anderson, Sarah York, Max Cant, Adam Cain, Adrian Bolton, Stephen Gaffney, Helen King, Demis Hassabis, Shane Legg, and Stig Petersen. Deepmind lab. *CoRR*, abs/1612.03801, 2016.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. *CoRR*, abs/1912.06680, 2019.

Maxime Chevalier-Boisvert. Miniworld: Minimalistic 3d environment for rl & robotics research. `https://github.com/maximecb/gym-miniworld`, 2018.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, Doha, Qatar, 2014.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, pages 2048–2056, 2020.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Conference of the Association for Computational*, pages 2978–2988, Florence, Italy, 2019.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de las Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, Jean-Marc Moret, Seb Noury, Federico Pesamosca, David Pfau, Olivier Sauter, Cristian Sommariva, Stefano Coda, Basil Duval, Ambrogio Fasoli, Pushmeet Kohli, Koray Kavukcuoglu, Demis Hassabis, and Martin Riedmiller. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602 (7897):414–419, 2022.

Rudi D'Hooge and Peter P De Deyn. Applications of the morris water maze in the study of learning and memory. *Brain research reviews*, 36(1):60–90, 2001.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep RL: A case study on PPO and TRPO. In *Proceedings of the 8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2020.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1406–1415, Stockholm, Sweden, 2018.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *Proceedings of the 35th Advances in Neural Information Processing Systems*, pages 18343–18362, New Orleans, Louisiana, 2022.

Meire Fortunato, Melissa Tan, Ryan Faulkner, Steven Hansen, Adrià Puigdomènech Badia, Gavin Buttimore, Charles Deck, Joel Z Leibo, and Charles Blundell. Generalization of reinforcement learners with working and episodic memory. In *Proceedings of the 32nd Advances in Neural Information Processing Systems*, pages 12448–12457, Vancouver, Canada, 2019.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

Matthew J. Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *Proceedings of the Association for the Advancement of Artificial Intelligence Fall Symposia, 2015*, pages 29–37, Arlington, Virginia, 2015.

Nicolas Heess, Jonathan J. Hunt, Timothy P. Lillicrap, and David Silver. Memory-based control with recurrent neural networks. *CoRR*, abs/1512.04455, 2015.

Felix Hill, Olivier Tieleman, Tamara von Glehn, Nathaniel Wong, Hamza Merzic, and Stephen Clark. Grounded language learning fast and slow. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. ICLR Blog Track, 2022a. URL `https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/`.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and JoÃ£o G.M. AraÃºjo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23 (274):1–18, 2022b. URL `http://jmlr.org/papers/v23/21-1342.html`.

Xiao Shi Huang, Felipe Pérez, Jimmy Ba, and Maksims Volkovs. Improving transformer optimization through better initialization. In *Proceedings of the 37th International Conference on Machine Learning*, pages 4475–4483, 2020.

Max Jaderberg, Wojciech M. Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio García Castañeda, Charles Beattie, Neil C. Rabinowitz, Ari S. Morcos, Avraham Ruderman, Nicolas Sonnerat, Tim Green, Louise Deason, Joel Z. Leibo, David Silver, Demis Hassabis, Koray Kavukcuoglu, and Thore Graepel. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *CoRR*, abs/1807.01281, 2018.

Scott M. Jordan, Adam White, Bruno Castro da Silva, Martha White, and Philip S. Thomas. Position: Benchmarking is limited in reinforcement learning research. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Steven Kapturowski, Georg Ostrovski, Will Dabney, John Quan, and Remi Munos. Recurrent experience replay in distributed reinforcement learning. In *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, Louisiana, 2019.

Andrew Kyle Lampinen, Stephanie C.Y. Chan, Andrea Banino, and Felix Hill. Towards mental time travel: a hierarchical memory for reinforcement learning agents. In *Proceedings of the 34th Advances in Neural Information Processing Systems*, pages 28182–28195, 2021.

Joel Z. Leibo, Cyprien de Masson d'Autume, Daniel Zoran, David Amos, Charles Beattie, Keith Anderson, Antonio García Castañeda, Manuel Sanchez, Simon Green, Audrunas Gruslys, Shane Legg, Demis Hassabis, and Matthew M. Botvinick. Psychlab: A psychology laboratory for deep reinforcement learning agents. *CoRR*, abs/1801.08116, 2018.

Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Nicolaus Foerster, Satinder Singh, and Feryal Behbahani. Structured state space models for in-context reinforcement learning. In *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems*, Honolulu, Hawaii, 2023.

Lingheng Meng, Rob Gorbet, and Dana Kulic. Memory-based deep reinforcement learning for pomdps. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5619–5626, Prague, Czech Republic, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1928–1937, New York City, New York, 2016.

Steven Morad, Ryan Kortvelesy, Matteo Bettini, Stephan Liwicki, and Amanda Prorok. POPGym: Benchmarking partially observable reinforcement learning. In *Proceedings of the 11th International Conference on Learning Representations*, Kigali, Ruanda, 2023.

Howard J. Morrison and Ralph H. Baer. Microcomputer controlled game, Sep 1977. URL: `https://patents.google.com/patent/US4207087A`.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

Fabian Paischer, Thomas Adler, Vihang P. Patil, Angela Bitto-Nemling, Markus Holzleitner, Sebastian Lehner, Hamid Eghbal-Zadeh, and Sepp Hochreiter. History compression via language models in reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning*, pages 17156–17185, Baltimore, Maryland, 2022a.

Fabian Paischer, Thomas Adler, Andreas Radler, Markus Hofmarcher, and Sepp Hochreiter. Toward semantic history compression for reinforcement learning. In *Language and Reinforcement Learning Workshop at Neural Information Processing Systems*, New Orleans, Louisiana, 2022b.

Emilio Parisotto and Ruslan Salakhutdinov. Efficient transformers in reinforcement learning using actor-learner distillation. In *Proceedings of the 9th International Conference on Learning Representation*, 2021.

Emilio Parisotto, H. Francis Song, Jack W. Rae, Razvan Pascanu, Çaglar Gülçehre, Siddhant M. Jayakumar, Max Jaderberg, Raphaël Lopez Kaufman, Aidan Clark, Seb Noury, Matthew M. Botvinick, Nicolas Heess, and Raia Hadsell. Stabilizing transformers for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, pages 7487–7498, 2020.

Jurgis Pašukonis, Timothy P Lillicrap, and Danijar Hafner. Evaluating long-term memory in 3d mazes. In *Proceedings of the 11th International Conference on Learning Representations*, Kigali, Ruanda, 2023.

Marco Pleines, Matthias Pallasch, Frank Zimmer, and Mike Preuss. Memory gym: Partially observable challenges to memory-based agents. In *Proceedings of the 11th International Conference on Learning Representations*, Kigali, Ruanda, 2023.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the 4th International Conference on Learning Representations*, San Juan, Puerto Rico, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

Wenling Shang, Xiaofei Wang, Aravind Srinivas, Aravind Rajeswaran, Yang Gao, Pieter Abbeel, and Michael Laskin. Reinforcement learning with latent flow. In *Proceedings of the 34th Advances in Neural Information Processing Systems*, pages 22171–22183, 2021.

H. Francis Song, Abbas Abdolmaleki, Jost Tobias Springenberg, Aidan Clark, Hubert Soyer, Jack W. Rae, Seb Noury, Arun Ahuja, Siqi Liu, Dhruva Tirumala, Nicolas Heess, Dan Belov, Martin A. Riedmiller, and Matthew M. Botvinick. V-MPO: on-policy maximum a posteriori policy optimization for discrete and continuous control. In *Proceedings of the 8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 30th Advances in Neural Information Processing Systems*, pages 5998–6008, Long Beach, California, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine,

Çaglar Gülçehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Rui Wang, Joel Lehman, Aditya Rawal, Jiale Zhi, Yulun Li, Jeffrey Clune, and Kenneth O. Stanley. Enhanced POET: open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In *Proceedings of the 37th International Conference on Machine Learning*, pages 9940–9951, 2020.

Daan Wierstra, Alexander Förster, Jan Peters, and Jürgen Schmidhuber. Solving deep memory pomdps with recurrent policy gradients. In *Proceedings of the 17th International Conference on Artificial Neural Networks*, pages 697–706, 2007.

Marek Wydmuch, Michał Kempka, and Wojciech Jaśkowski. Vizdoom competitions: Playing doom from pixels. *IEEE Transactions on Games*, 11(3):248–259, 2019.